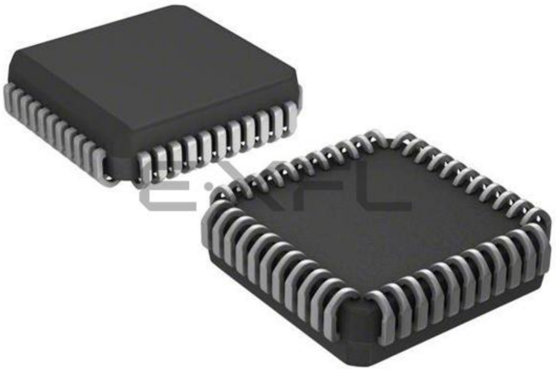


What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	25MHz
Connectivity	UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	33
Program Memory Size	4KB (2K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	232 x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 6V
Data Converters	-
Oscillator Type	External
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	44-PLCC (16.59x16.59)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic17c42a-25-l

6.0 MEMORY ORGANIZATION

There are two memory blocks in the PIC17C4X; program memory and data memory. Each block has its own bus, so that access to each block can occur during the same oscillator cycle.

The data memory can further be broken down into General Purpose RAM and the Special Function Registers (SFRs). The operation of the SFRs that control the “core” are described here. The SFRs used to control the peripheral modules are described in the section discussing each individual peripheral module.

6.1 Program Memory Organization

PIC17C4X devices have a 16-bit program counter capable of addressing a 64K x 16 program memory space. The reset vector is at 0000h and the interrupt vectors are at 0008h, 0010h, 0018h, and 0020h (Figure 6-1).

6.1.1 PROGRAM MEMORY OPERATION

The PIC17C4X can operate in one of four possible program memory configurations. The configuration is selected by two configuration bits. The possible modes are:

- Microprocessor
- Microcontroller
- Extended Microcontroller
- Protected Microcontroller

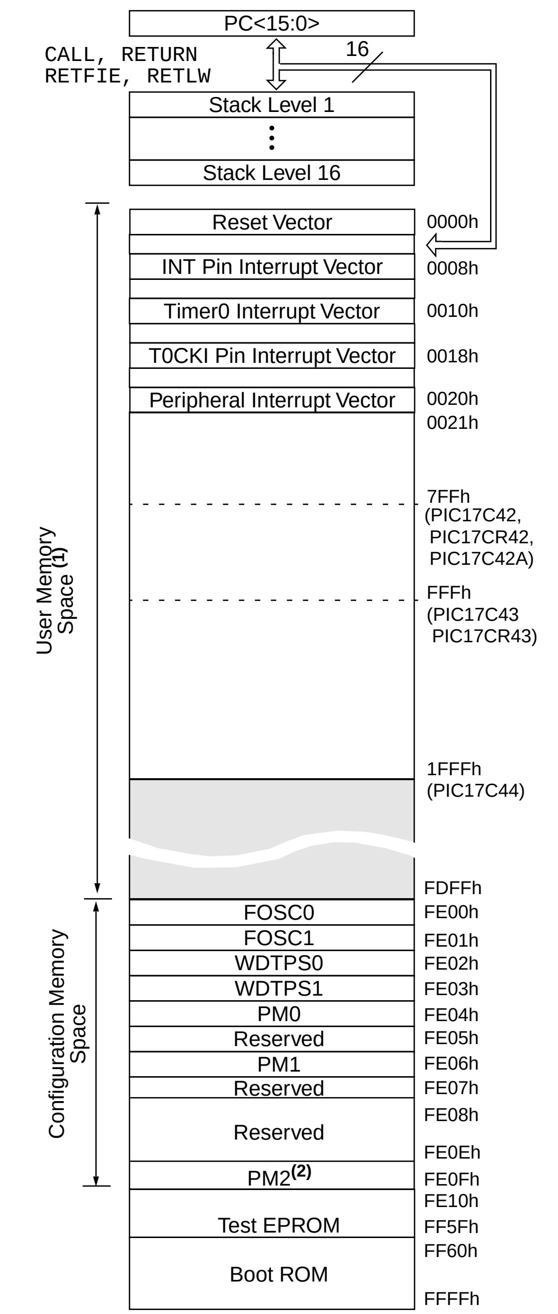
The microcontroller and protected microcontroller modes only allow internal execution. Any access beyond the program memory reads unknown data. The protected microcontroller mode also enables the code protection feature.

The extended microcontroller mode accesses both the internal program memory as well as external program memory. Execution automatically switches between internal and external memory. The 16-bits of address allow a program memory range of 64K-words.

The microprocessor mode only accesses the external program memory. The on-chip program memory is ignored. The 16-bits of address allow a program memory range of 64K-words. Microprocessor mode is the default mode of an unprogrammed device.

The different modes allow different access to the configuration bits, test memory, and boot ROM. Table 6-1 lists which modes can access which areas in memory. Test Memory and Boot Memory are not required for normal operation of the device. Care should be taken to ensure that no unintended branches occur to these areas.

FIGURE 6-1: PROGRAM MEMORY MAP AND STACK



Note 1: User memory space may be internal, external, or both. The memory configuration depends on the processor mode.

2: This location is reserved on the PIC17C42.

6.2.2.2 CPU STATUS REGISTER (CPUSTA)

The CPUSTA register contains the status and control bits for the CPU. This register is used to globally enable/disable interrupts. If only a specific interrupt is desired to be enabled/disabled, please refer to the INTerrupt STAtus (INTSTA) register and the Peripheral Interrupt Enable (PIE) register. This register also indicates if the stack is available and contains the Power-down ($\overline{PD}$) and Time-out ($\overline{TO}$) bits. The $\overline{TO}$, $\overline{PD}$, and STKAV bits are not writable. These bits are set and cleared according to device logic. Therefore, the result of an instruction with the CPUSTA register as destination may be different than intended.

FIGURE 6-8: CPUSTA REGISTER (ADDRESS: 06h, UNBANKED)

U - 0	U - 0	R - 1	R/W - 1	R - 1	R - 1	U - 0	U - 0
—	—	STKAV	GLINTD	$\overline{TO}$	$\overline{PD}$	—	—
bit7							bit0

R = Readable bit
W = Writable bit
U = Unimplemented bit, Read as ‘0’
- n = Value at POR reset

bit 7-6: **Unimplemented**: Read as '0'

bit 5: **STKAV**: Stack Available bit
This bit indicates that the 4-bit stack pointer value is Fh, or has rolled over from Fh → 0h (stack overflow).
1 = Stack is available
0 = Stack is full, or a stack overflow may have occurred (Once this bit has been cleared by a stack overflow, only a device reset will set this bit)

bit 4: **GLINTD**: Global Interrupt Disable bit
This bit disables all interrupts. When enabling interrupts, only the sources with their enable bits set can cause an interrupt.
1 = Disable all interrupts
0 = Enables all un-masked interrupts

bit 3: **$\overline{TO}$**: WDT Time-out Status bit
1 = After power-up or by a `CLRWDT` instruction
0 = A Watchdog Timer time-out occurred

bit 2: **$\overline{PD}$**: Power-down Status bit
1 = After power-up or by the `CLRWDT` instruction
0 = By execution of the `SLEEP` instruction

bit 1-0: **Unimplemented**: Read as '0'

7.0 TABLE READS AND TABLE WRITES

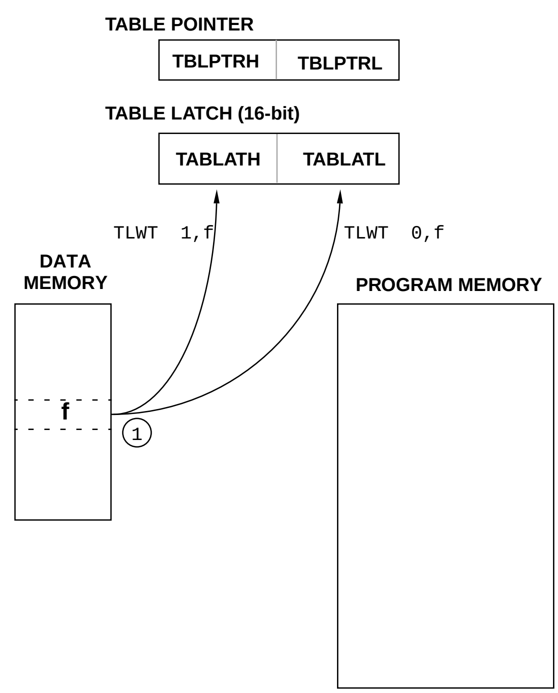
The PIC17C4X has four instructions that allow the processor to move data from the data memory space to the program memory space, and vice versa. Since the program memory space is 16-bits wide and the data memory space is 8-bits wide, two operations are required to move 16-bit values to/from the data memory.

The `TLWT t,f` and `TABLWT t,i,f` instructions are used to write data from the data memory space to the program memory space. The `TLRD t,f` and `TABLRD t,i,f` instructions are used to write data from the program memory space to the data memory space.

The program memory can be internal or external. For the program memory access to be external, the device needs to be operating in extended microcontroller or microprocessor mode.

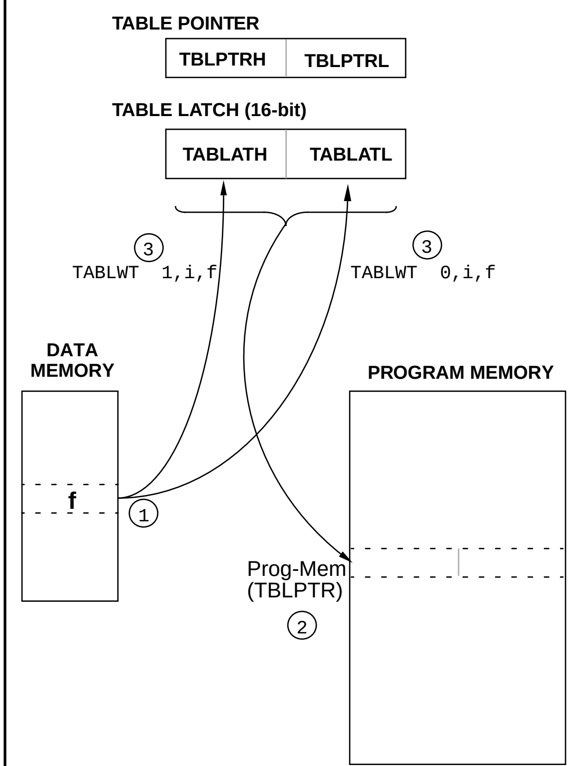
Figure 7-1 through Figure 7-4 show the operation of these four instructions.

FIGURE 7-1: TLWT INSTRUCTION OPERATION



Note 1: 8-bit value, from register 'f', loaded into the high or low byte in TABLAT (16-bit).

FIGURE 7-2: TABLWT INSTRUCTION OPERATION



Note 1: 8-bit value, from register 'f', loaded into the high or low byte in TABLAT (16-bit).
2: 16-bit TABLAT value written to address Program Memory (TBLPTR).
3: If “i” = 1, then TBLPTR = TBLPTR + 1,
If “i” = 0, then TBLPTR is unchanged.

Example 8-4 shows the sequence to do an 16 x 16 signed multiply. Equation 8-2 shows the algorithm that used. The 32-bit result is stored in four registers RES3:RES0. To account for the sign bits of the arguments, each argument pairs most significant bit (MSb) is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

RES3:RES0

= ARG1H:ARG1L * ARG2H:ARG2L

= (ARG1H * ARG2H * 2^{16}) +

(ARG1H * ARG2L * 2^{8}) +

(ARG1L * ARG2H * 2^{8}) +

(ARG1L * ARG2L) +

(-1 * ARG2H<7> * ARG1H:ARG1L * 2^{16}) +

(-1 * ARG1H<7> * ARG2H:ARG2L * 2^{16})

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
   MOVFP    ARG1L, WREG
   MULWF    ARG2L       ; ARG1L * ARG2L ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES1 ;
   MOVPF    PRODL, RES0 ;
;
   MOVFP    ARG1H, WREG
   MULWF    ARG2H       ; ARG1H * ARG2H ->
                        ;   PRODH:PRODL
   MOVPF    PRODH, RES3 ;
   MOVPF    PRODL, RES2 ;
;
   MOVFP    ARG1L, WREG
   MULWF    ARG2H       ; ARG1L * ARG2H ->
                        ;   PRODH:PRODL
   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   MOVFP    ARG1H, WREG ;
   MULWF    ARG2L       ; ARG1H * ARG2L ->
                        ;   PRODH:PRODL

   MOVFP    PRODL, WREG ;
   ADDWF    RES1, F     ; Add cross
   MOVFP    PRODH, WREG ;    products
   ADDWFC   RES2, F     ;
   CLRF     WREG, F     ;
   ADDWFC   RES3, F     ;
;
   BTFSS    ARG2H, 7    ; ARG2H:ARG2L neg?
   GOTO     SIGN_ARG1   ; no, check ARG1
   MOVFP    ARG1L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG1H, WREG ;
   SUBWFB   RES3
;
SIGN_ARG1
   BTFSS    ARG1H, 7    ; ARG1H:ARG1L neg?
   GOTO     CONT_CODE   ; no, done
   MOVFP    ARG2L, WREG ;
   SUBWF    RES2        ;
   MOVFP    ARG2H, WREG ;
   SUBWFB   RES3
;
CONT_CODE
      :
```

TABLE 9-9: PORTE FUNCTIONS

Name	Bit	Buffer Type	Function
RE0/ALE	bit0	TTL	Input/Output or system bus Address Latch Enable (ALE) control pin.
RE1/$\overline{OE}$	bit1	TTL	Input/Output or system bus Output Enable ($\overline{OE}$) control pin.
RE2/$\overline{WR}$	bit2	TTL	Input/Output or system bus Write ($\overline{WR}$) control pin.

Legend: TTL = TTL input.

TABLE 9-10: REGISTERS/BITS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on Power-on Reset	Value on all other resets (Note1)
15h, Bank 1	PORTE	—	—	—	—	—	RE2/$\overline{WR}$	RE1/$\overline{OE}$	RE0/ALE	---- -xxx	---- -uuu
14h, Bank 1	DDRE	Data direction register for PORTE								---- -111	---- -111

Legend: x = unknown, u = unchanged, - = unimplemented read as '0'. Shaded cells are not used by PORTE.
Note 1: Other (non power-up) resets include: external reset through $\overline{MCLR}$ and the Watchdog Timer Reset.

9.5 I/O Programming Considerations

9.5.1 BI-DIRECTIONAL I/O PORTS

Any instruction which writes, operates internally as a read followed by a write operation. For example, the BCF and BSF instructions read the register into the CPU, execute the bit operation, and write the result back to the register. Caution must be used when these instructions are applied to a port with both inputs and outputs defined. For example, a BSF operation on bit5 of PORTB will cause all eight bits of PORTB to be read into the CPU. Then the BSF operation takes place on bit5 and PORTB is written to the output latches. If another bit of PORTB is used as a bi-directional I/O pin (e.g. bit0) and it is defined as an input at this time, the input signal present on the pin itself would be read into the CPU and re-written to the data latch of this particular pin, overwriting the previous content. As long as the pin stays in the input mode, no problem occurs. However, if bit0 is switched into output mode later on, the content of the data latch may now be unknown.

Reading a port reads the values of the port pins. Writing to the port register writes the value to the port latch. When using read-modify-write instructions (BCF, BSF, BTG, etc.) on a port, the value of the port pins is read, the desired operation is performed with this value, and the value is then written to the port latch.

Example 9-5 shows the effect of two sequential read-modify-write instructions on an I/O port.

EXAMPLE 9-5: READ MODIFY WRITE INSTRUCTIONS ON AN I/O PORT

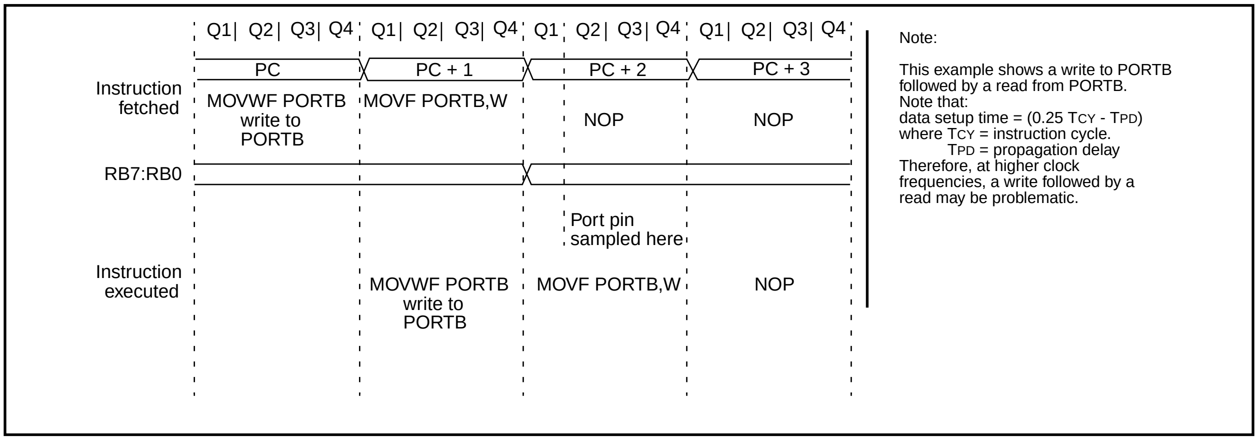
```
; Initial PORT settings: PORTB<7:4> Inputs
;                        PORTB<3:0> Outputs
; PORTB<7:6> have pull-ups and are
; not connected to other circuitry
;
;                        PORT latch  PORT pins
;                        ----------  ---------
;
  BCF   PORTB, 7         01pp pppp   11pp pppp
  BCF   PORTB, 6         10pp pppp   11pp pppp
;
  BCF   DDRB, 7          10pp pppp   11pp pppp
  BCF   DDRB, 6          10pp pppp   10pp pppp
;
; Note that the user may have expected the
; pin values to be 00pp pppp. The 2nd BCF
; caused RB7 to be latched as the pin value
; (High).
```

> **Note:** A pin actively outputting a Low or High should not be driven from external devices in order to change the level on this pin (i.e. “wired-or”, “wired-and”). The resulting high output currents may damage the device.

9.5.2 SUCCESSIVE OPERATIONS ON I/O PORTS

The actual write to an I/O port happens at the end of an instruction cycle, whereas for reading, the data must be valid at the beginning of the instruction cycle (Figure 9-9). Therefore, care must be exercised if a write followed by a read operation is carried out on the same I/O port. The sequence of instructions should be such to allow the pin voltage to stabilize (load dependent) before executing the instruction that reads the values on that I/O port. Otherwise, the previous state of that pin may be read into the CPU rather than the “new” state. When in doubt, it is better to separate these instructions with a NOP or another instruction not accessing this I/O port.

FIGURE 9-9: SUCCESSIVE I/O OPERATION



11.1 Timer0 Operation

When the T0CS (T0STA<5>) bit is set, TMR0 increments on the internal clock. When T0CS is clear, TMR0 increments on the external clock (RA1/T0CKI pin). The external clock edge can be configured in software. When the T0SE (T0STA<6>) bit is set, the timer will increment on the rising edge of the RA1/T0CKI pin. When T0SE is clear, the timer will increment on the falling edge of the RA1/T0CKI pin. The prescaler can be programmed to introduce a prescale of 1:1 to 1:256. The timer increments from 0000h to FFFFh and rolls over to 0000h. On overflow, the TMR0 Interrupt Flag bit (T0IF) is set. The TMR0 interrupt can be masked by clearing the corresponding TMR0 Interrupt Enable bit (T0IE). The TMR0 Interrupt Flag bit (T0IF) is automatically cleared when vectoring to the TMR0 interrupt vector.

11.2 Using Timer0 with External Clock

When the external clock input is used for Timer0, it is synchronized with the internal phase clocks. Figure 11-3 shows the synchronization of the external clock. This synchronization is done after the prescaler. The output of the prescaler (PSOUT) is sampled twice in every instruction cycle to detect a rising or a falling edge. The timing requirements for the external clock are detailed in the electrical specification section for the desired device.

11.2.1 DELAY FROM EXTERNAL CLOCK EDGE

Since the prescaler output is synchronized with the internal clocks, there is a small delay from the time the external clock edge occurs to the time TMR0 is actually incremented. Figure 11-3 shows that this delay is between 3TOSC and 7TOSC. Thus, for example, measuring the interval between two edges (e.g. period) will be accurate within ±4TOSC (±121 ns @ 33 MHz).

FIGURE 11-2: TIMER0 MODULE BLOCK DIAGRAM

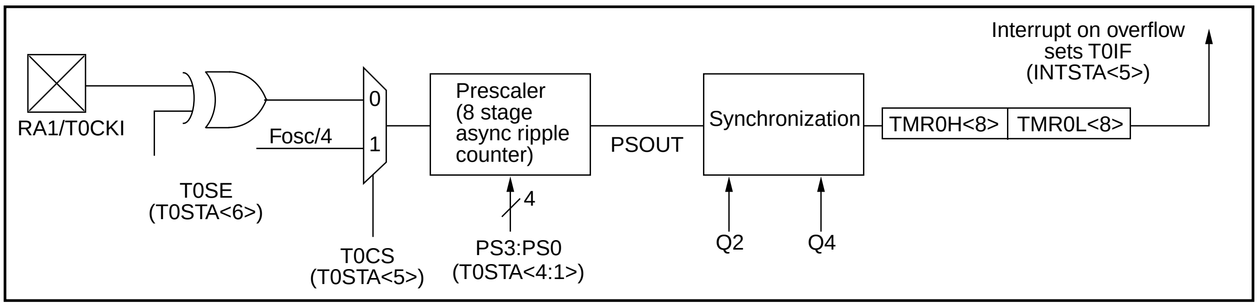
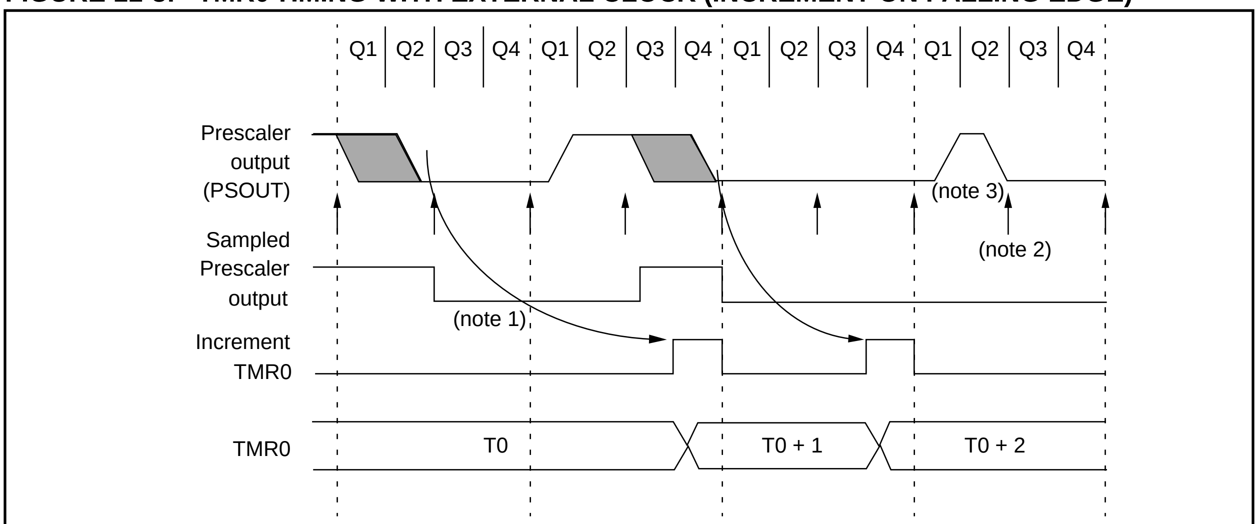


FIGURE 11-3: TMR0 TIMING WITH EXTERNAL CLOCK (INCREMENT ON FALLING EDGE)



Note 1: The delay from the T0CKI edge to the TMR0 increment is 3Tosc to 7Tosc.
2: ↑ = PSOUT is sampled here.
3: The PSOUT high time is too short and is missed by the sampling circuit.

12.1 Timer1 and Timer2

12.1.1 TIMER1, TIMER2 IN 8-BIT MODE

Both Timer1 and Timer2 will operate in 8-bit mode when the T16 bit is clear. These two timers can be independently configured to increment from the internal instruction cycle clock or from an external clock source on the RB4/TCLK12 pin. The timer clock source is configured by the TMRxCS bit (x = 1 for Timer1 or = 2 for Timer2). When TMRxCS is clear, the clock source is internal and increments once every instruction cycle (Fosc/4). When TMRxCS is set, the clock source is the RB4/TCLK12 pin, and the timer will increment on every falling edge of the RB4/TCLK12 pin.

The timer increments from 00h until it equals the Period register (PRx). It then resets to 00h at the next increment cycle. The timer interrupt flag is set when the timer is reset. TMR1 and TMR2 have individual interrupt flag bits. The TMR1 interrupt flag bit is latched into TMR1IF, and the TMR2 interrupt flag bit is latched into TMR2IF.

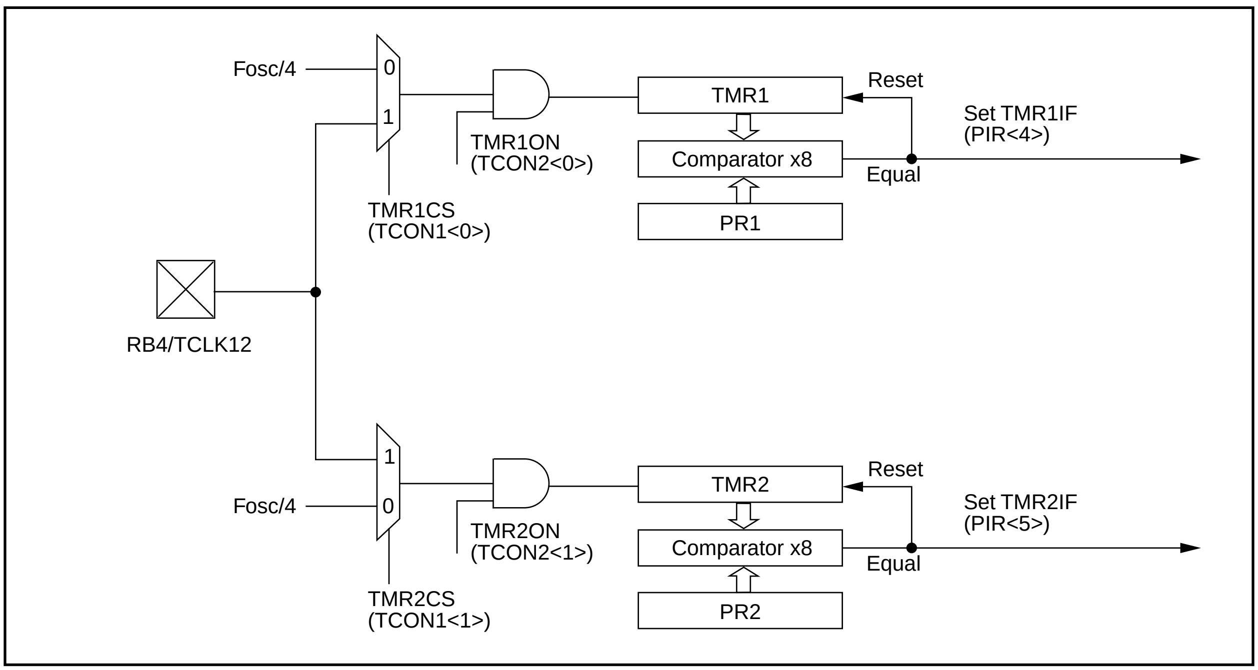
Each timer also has a corresponding interrupt enable bit (TMRxIE). The timer interrupt can be enabled by setting this bit and disabled by clearing this bit. For peripheral interrupts to be enabled, the Peripheral Interrupt Enable bit must be enabled (PEIE is set) and global interrupts must be enabled (GLINTD is cleared).

The timers can be turned on and off under software control. When the Timerx On control bit (TMRxON) is set, the timer increments from the clock source. When TMRxON is cleared, the timer is turned off and cannot cause the timer interrupt flag to be set.

12.1.1.1 EXTERNAL CLOCK INPUT FOR TIMER1 OR TIMER2

When TMRxCS is set, the clock source is the RB4/TCLK12 pin, and the timer will increment on every falling edge on the RB4/TCLK12 pin. The TCLK12 input is synchronized with internal phase clocks. This causes a delay from the time a falling edge appears on TCLK12 to the time TMR1 or TMR2 is actually incremented. For the external clock input timing requirements, see the Electrical Specification section.

FIGURE 12-3: TIMER1 AND TIMER2 IN TWO 8-BIT TIMER/COUNTER MODE



14.2.4 EXTERNAL CRYSTAL OSCILLATOR CIRCUIT

Either a prepackaged oscillator can be used or a simple oscillator circuit with TTL gates can be built. Prepackaged oscillators provide a wide operating range and better stability. A well-designed crystal oscillator will provide good performance with TTL gates. Two types of crystal oscillator circuits can be used: one with series resonance, or one with parallel resonance.

Figure 14-5 shows implementation of a parallel resonant oscillator circuit. The circuit is designed to use the fundamental frequency of the crystal. The 74AS04 inverter performs the 180-degree phase shift that a parallel oscillator requires. The 4.7 kΩ resistor provides the negative feedback for stability. The 10 kΩ potentiometer biases the 74AS04 in the linear region. This could be used for external oscillator designs.

FIGURE 14-5: EXTERNAL PARALLEL RESONANT CRYSTAL OSCILLATOR CIRCUIT

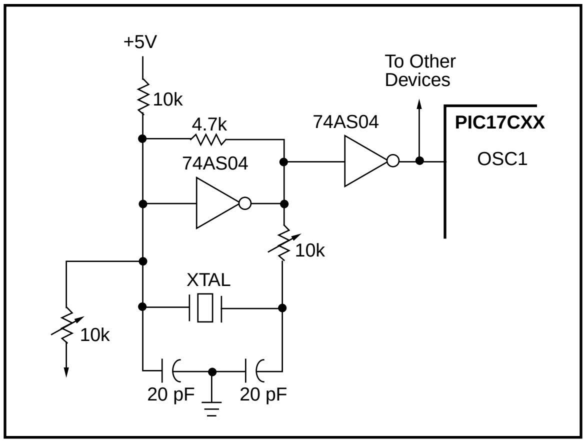
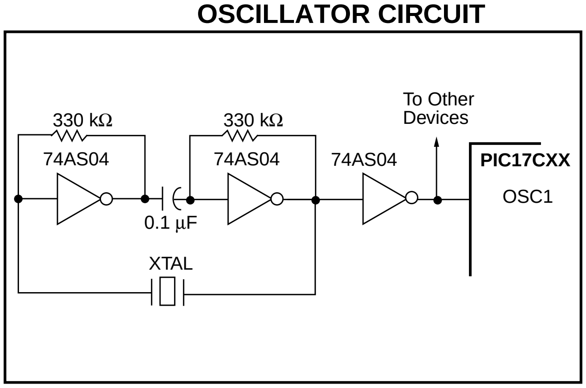


Figure 14-6 shows a series resonant oscillator circuit. This circuit is also designed to use the fundamental frequency of the crystal. The inverter performs a 180-degree phase shift in a series resonant oscillator circuit. The 330 kΩ resistors provide the negative feedback to bias the inverters in their linear region.

FIGURE 14-6: EXTERNAL SERIES RESONANT CRYSTAL OSCILLATOR CIRCUIT



14.2.5 RC OSCILLATOR

For timing insensitive applications, the RC device option offers additional cost savings. RC oscillator frequency is a function of the supply voltage, the resistor (Rext) and capacitor (Cext) values, and the operating temperature. In addition to this, oscillator frequency will vary from unit to unit due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect oscillation frequency, especially for low Cext values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 14-6 shows how the R/C combination is connected to the PIC17CXX. For Rext values below 2.2 kΩ, the oscillator operation may become unstable, or stop completely. For very high Rext values (e.g. 1 MΩ), the oscillator becomes sensitive to noise, humidity and leakage. Thus, we recommend to keep Rext between 3 kΩ and 100 kΩ.

Although the oscillator will operate with no external capacitor (Cext = 0 pF), we recommend using values above 20 pF for noise and stability reasons. With little or no external capacitance, oscillation frequency can vary dramatically due to changes in external capacitances, such as PCB trace capacitance or package lead frame capacitance.

See Section 18.0 for RC frequency variation from part to part due to normal process variation. The variation is larger for larger R (since leakage current variation will affect RC frequency more for large R) and for smaller C (since variation of input capacitance will affect RC frequency more).

See Section 18.0 for variation of oscillator frequency due to VDD for given Rext/Cext values as well as frequency variation due to operating temperature for given R, C, and VDD values.

The oscillator frequency, divided by 4, is available on the OSC2/CLKOUT pin, and can be used for test purposes or to synchronize other logic (see Figure 3-2 for waveform).

FIGURE 14-7: RC OSCILLATOR MODE

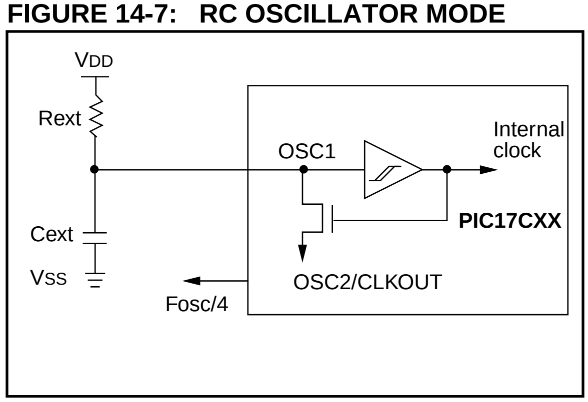


TABLE 15-2: PIC17CXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-bit Opcode MSb … LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	**f,d**	ADD WREG to f	1	`0000 111d ffff ffff`	OV,C,DC,Z	
ADDWFC	**f,d**	ADD WREG and Carry bit to f	1	`0001 000d ffff ffff`	OV,C,DC,Z	
ANDWF	**f,d**	AND WREG with f	1	`0000 101d ffff ffff`	Z	
CLRF	**f,s**	Clear f, or Clear f and Clear WREG	1	`0010 100s ffff ffff`	None	3
COMF	**f,d**	Complement f	1	`0001 001d ffff ffff`	Z	
CPFSEQ	**f**	Compare f with WREG, skip if f = WREG	1 (2)	`0011 0001 ffff ffff`	None	6,8
CPFSGT	**f**	Compare f with WREG, skip if f > WREG	1 (2)	`0011 0010 ffff ffff`	None	2,6,8
CPFSLT	**f**	Compare f with WREG, skip if f < WREG	1 (2)	`0011 0000 ffff ffff`	None	2,6,8
DAW	**f,s**	Decimal Adjust WREG Register	1	`0010 111s ffff ffff`	C	3
DECF	**f,d**	Decrement f	1	`0000 011d ffff ffff`	OV,C,DC,Z	
DECFSZ	**f,d**	Decrement f, skip if 0	1 (2)	`0001 011d ffff ffff`	None	6,8
DCFSNZ	**f,d**	Decrement f, skip if not 0	1 (2)	`0010 011d ffff ffff`	None	6,8
INCF	**f,d**	Increment f	1	`0001 010d ffff ffff`	OV,C,DC,Z	
INCFSZ	**f,d**	Increment f, skip if 0	1 (2)	`0001 111d ffff ffff`	None	6,8
INFSNZ	**f,d**	Increment f, skip if not 0	1 (2)	`0010 010d ffff ffff`	None	6,8
IORWF	**f,d**	Inclusive OR WREG with f	1	`0000 100d ffff ffff`	Z	
MOVFP	**f,p**	Move f to p	1	`011p pppp ffff ffff`	None	
MOVPF	**p,f**	Move p to f	1	`010p pppp ffff ffff`	Z	
MOVWF	**f**	Move WREG to f	1	`0000 0001 ffff ffff`	None	
MULWF	**f**	Multiply WREG with f	1	`0011 0100 ffff ffff`	None	9
NEGW	**f,s**	Negate WREG	1	`0010 110s ffff ffff`	OV,C,DC,Z	1,3
NOP	**—**	No Operation	1	`0000 0000 0000 0000`	None	
RLCF	**f,d**	Rotate left f through Carry	1	`0001 101d ffff ffff`	C	
RLNCF	**f,d**	Rotate left f (no carry)	1	`0010 001d ffff ffff`	None	
RRCF	**f,d**	Rotate right f through Carry	1	`0001 100d ffff ffff`	C	
RRNCF	**f,d**	Rotate right f (no carry)	1	`0010 000d ffff ffff`	None	
SETF	**f,s**	Set f	1	`0010 101s ffff ffff`	None	3
SUBWF	**f,d**	Subtract WREG from f	1	`0000 010d ffff ffff`	OV,C,DC,Z	1
SUBWFB	**f,d**	Subtract WREG from f with Borrow	1	`0000 001d ffff ffff`	OV,C,DC,Z	1
SWAPF	**f,d**	Swap f	1	`0001 110d ffff ffff`	None	
TABLRD	**t,i,f**	Table Read	2 (3)	`1010 10ti ffff ffff`	None	7

Legend: Refer to Table 15-1 for opcode field descriptions.

Note 1: 2's Complement method.

2: Unsigned arithmetic.

3: If s = '1', only the file is affected: If s = '0', both the WREG register and the file are affected; If only the Working register (WREG) is required to be affected, then f = WREG must be specified.

4: During an `LCALL`, the contents of PCLATH are loaded into the MSB of the PC and `kkkk kkkk` is loaded into the LSB of the PC (PCL)

5: Multiple cycle instruction for EPROM programming when table pointer selects internal EPROM. The instruction is terminated by an interrupt event. When writing to external program memory, it is a two-cycle instruction.

6: Two-cycle instruction when condition is true, else single cycle instruction.

7: Two-cycle instruction except for `TABLRD` to PCL (program counter low byte) in which case it takes 3 cycles.

8: A "skip" means that instruction fetched during execution of current instruction is not executed, instead an NOP is executed.

9: These instructions are not available on the PIC17C42.

CALL **Subroutine Call**

Syntax: [*label*] CALL k

Operands: $0 \le k \le 4095$

Operation: PC+ 1→ TOS, k → PC<12:0>, k<12:8> → PCLATH<4:0>; PC<15:13> → PCLATH<7:5>

Status Affected: None

Encoding:

111k	kkkk	kkkk	kkkk

Description: Subroutine call within 8K page. First, return address (PC+1) is pushed onto the stack. The 13-bit value is loaded into PC bits<12:0>. Then the upper-eight bits of the PC are copied into PCLATH. `Call` is a two-cycle instruction.

See `LCALL` for calls outside 8K memory space.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>	Execute	NOP
Forced NOP	NOP	Execute	NOP

Example: `HERE CALL THERE`

Before Instruction
PC = Address(`HERE`)

After Instruction
PC = Address(`THERE`)
TOS = Address (`HERE + 1`)

CLRF **Clear f**

Syntax: [*label*] CLRF f,s

Operands: $0 \le f \le 255$

Operation: 00h → f, s ∈ [0,1]
00h → dest

Status Affected: None

Encoding:

0010	100s	ffff	ffff

Description: Clears the contents of the specified register(s).
s = 0: Data memory location 'f' and WREG are cleared.
s = 1: Data memory location 'f' is cleared.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Execute	Write register 'f' and other specified register

Example: `CLRF FLAG_REG`

Before Instruction
FLAG_REG = 0x5A

After Instruction
FLAG_REG = 0x00

MOVLR — Move Literal to high nibble in BSR

Syntax:	[*label*] MOVLR k
Operands:	0 ≤ k ≤ 15
Operation:	k → (BSR<7:4>)
Status Affected:	None
Encoding:	`1011` `101x` `kkkk` `uuuu`
Description:	The 4-bit literal 'k' is loaded into the most significant 4-bits of the Bank Select Register (BSR). Only the high 4-bits of the Bank Select Register are affected. The lower half of the BSR is unchanged. The assembler will encode the “u” fields as 0.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k:u'	Execute	Write literal 'k' to BSR<7:4>

Example: `MOVLR 5`

Before Instruction
BSR register = 0x22

After Instruction
BSR register = 0x52

Note: This instruction is not available in the PIC17C42 device.

MOVLW — Move Literal to WREG

Syntax:	[*label*] MOVLW k
Operands:	0 ≤ k ≤ 255
Operation:	k → (WREG)
Status Affected:	None
Encoding:	`1011` `0000` `kkkk` `kkkk`
Description:	The eight bit literal 'k' is loaded into WREG.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Execute	Write to WREG

Example: `MOVLW 0x5A`

After Instruction
WREG = 0x5A

TABLWT **Table Write**

Example1: `TABLWT 0, 1, REG`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0xFFFF

After Instruction (table write completion)

REG	=	0x53
TBLATH	=	0x53
TBLATL	=	0x55
TBLPTR	=	0xA357
MEMORY(TBLPTR - 1)	=	0x5355

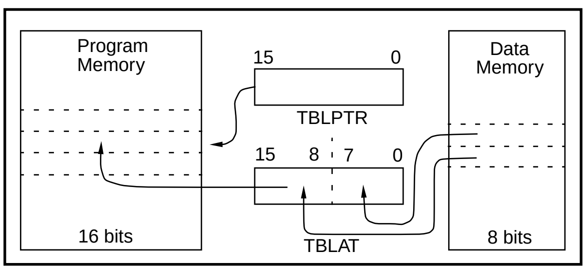
Example 2: `TABLWT 1, 0, REG`

Before Instruction

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x55
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0xFFFF

After Instruction (table write completion)

REG	=	0x53
TBLATH	=	0xAA
TBLATL	=	0x53
TBLPTR	=	0xA356
MEMORY(TBLPTR)	=	0xAA53



TLRD **Table Latch Read**

Syntax: [*label*] TLRD t,f

Operands: 0 ≤ f ≤ 255
t ∈ [0,1]

Operation: If t = 0,
TBLATL → f;
If t = 1,
TBLATH → f

Status Affected: None

Encoding:

1010	00tx	ffff	ffff

Description: Read data from 16-bit table latch (TBLAT) into file register 'f'. Table Latch is unaffected.

If t = 1; high byte is read

If t = 0; low byte is read

This instruction is used in conjunction with `TABLRD` to transfer data from program memory to data memory.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register TBLATH or TBLATL	Execute	Write register 'f'

Example: `TLRD t, RAM`

Before Instruction

t	=	0	
RAM	=	?	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)

After Instruction

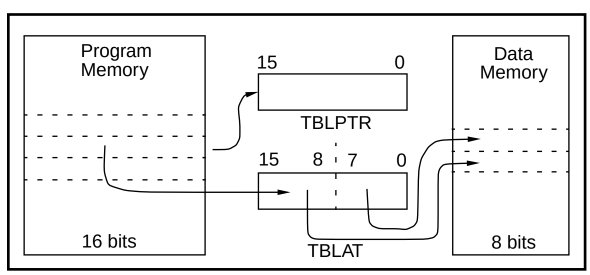
RAM	=	0xAF	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)

Before Instruction

t	=	1	
RAM	=	?	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)

After Instruction

RAM	=	0x00	
TBLAT	=	0x00AF	(TBLATH = 0x00) (TBLATL = 0xAF)



Applicable Devices	42	R42	42A	43	R43	44

TABLE 17-1: CROSS REFERENCE OF DEVICE SPECS FOR OSCILLATOR CONFIGURATIONS AND FREQUENCIES OF OPERATION (COMMERCIAL DEVICES)

OSC	PIC17C42-16	PIC17C42-25
RC	V_{DD}: 4.5V to 5.5V I_{DD}: 6 mA max. I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 4 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 6 mA max. I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 4 MHz max.
XT	V_{DD}: 4.5V to 5.5V I_{DD}: 24 mA max. I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 16 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 25 MHz max.
EC	V_{DD}: 4.5V to 5.5V I_{DD}: 24 mA max. I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 16 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 38 mA max. I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 25 MHz max.
LF	V_{DD}: 4.5V to 5.5V I_{DD}: 150 µA max. at 32 kHz (WDT enabled) I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 2 MHz max.	V_{DD}: 4.5V to 5.5V I_{DD}: 150 µA max. at 32 kHz (WDT enabled) I_{PD}: 5 µA max. at 5.5V (WDT disabled) Freq: 2 MHz max.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 18-5: TRANSCONDUCTANCE (gm) OF LF OSCILLATOR vs. V_{DD}

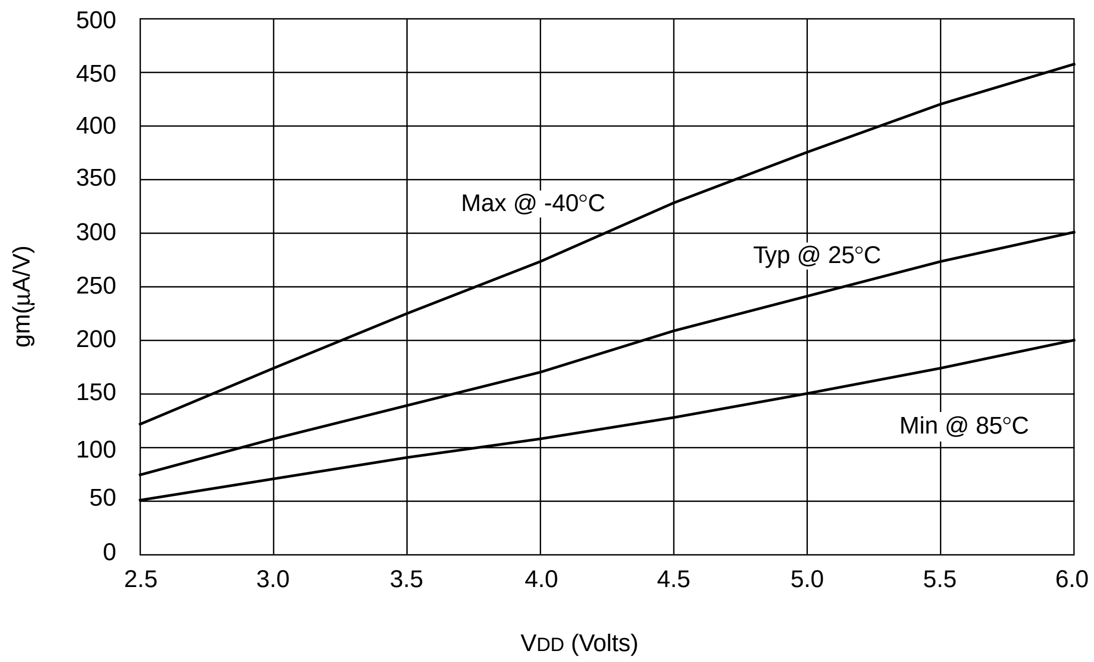
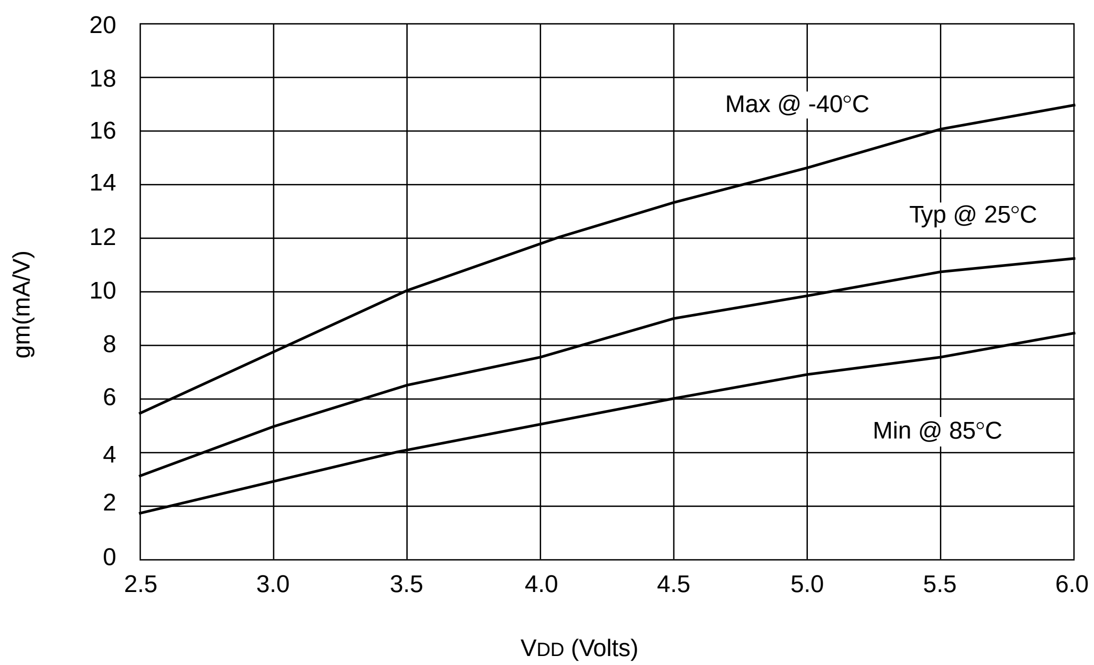


FIGURE 18-6: TRANSCONDUCTANCE (gm) OF XT OSCILLATOR vs. V_{DD}



Applicable Devices	42	R42	42A	43	R43	44

19.5 Timing Diagrams and Specifications

FIGURE 19-2: EXTERNAL CLOCK TIMING

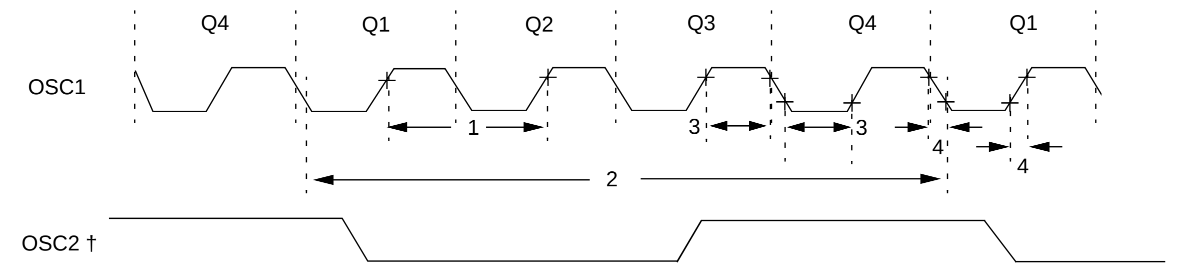


TABLE 19-2: EXTERNAL CLOCK TIMING REQUIREMENTS

Param No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
	Fosc	**External CLKIN Frequency (Note 1)**	DC	—	8	MHz	EC osc mode - 08 devices (8 MHz devices)
			DC	—	16	MHz	- 16 devices (16 MHz devices)
			DC	—	25	MHz	- 25 devices (25 MHz devices)
			DC	—	33	MHz	- 33 devices (33 MHz devices)
		Oscillator Frequency (Note 1)	DC	—	4	MHz	RC osc mode
			1	—	8	MHz	XT osc mode - 08 devices (8 MHz devices)
			1	—	16	MHz	- 16 devices (16 MHz devices)
			1	—	25	MHz	- 25 devices (25 MHz devices)
			1	—	33	MHz	- 33 devices (33 MHz devices)
			DC	—	2	MHz	LF osc mode
1	Tosc	**External CLKIN Period (Note 1)**	125	—	—	ns	EC osc mode - 08 devices (8 MHz devices)
			62.5	—	—	ns	- 16 devices (16 MHz devices)
			40	—	—	ns	- 25 devices (25 MHz devices)
			30.3	—	—	ns	- 33 devices (33 MHz devices)
		Oscillator Period (Note 1)	250	—	—	ns	RC osc mode
			125	—	1,000	ns	XT osc mode - 08 devices (8 MHz devices)
			62.5	—	1,000	ns	- 16 devices (16 MHz devices)
			40	—	1,000	ns	- 25 devices (25 MHz devices)
			30.3	—	1,000	ns	- 33 devices (33 MHz devices)
			500	—	—	ns	LF osc mode
2	TCY	**Instruction Cycle Time (Note 1)**	121.2	4/Fosc	DC	ns	
3	TosL, TosH	**Clock in (OSC1) high or low time**	10 ‡	—	—	ns	EC oscillator
4	TosR, TosF	**Clock in (OSC1) rise or fall time**	—	—	5 ‡	ns	EC oscillator

† Data in “Typ” column is at 5V, 25˚C unless otherwise stated. These parameters are for design guidance only and are not tested.

‡ These parameters are for design guidance only and are not tested, nor characterized.

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKIN pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

Applicable Devices	42	R42	42A	43	R43	44

FIGURE 20-19: VTH, VIL of I/O PINS (SCHMITT TRIGGER) vs. VDD

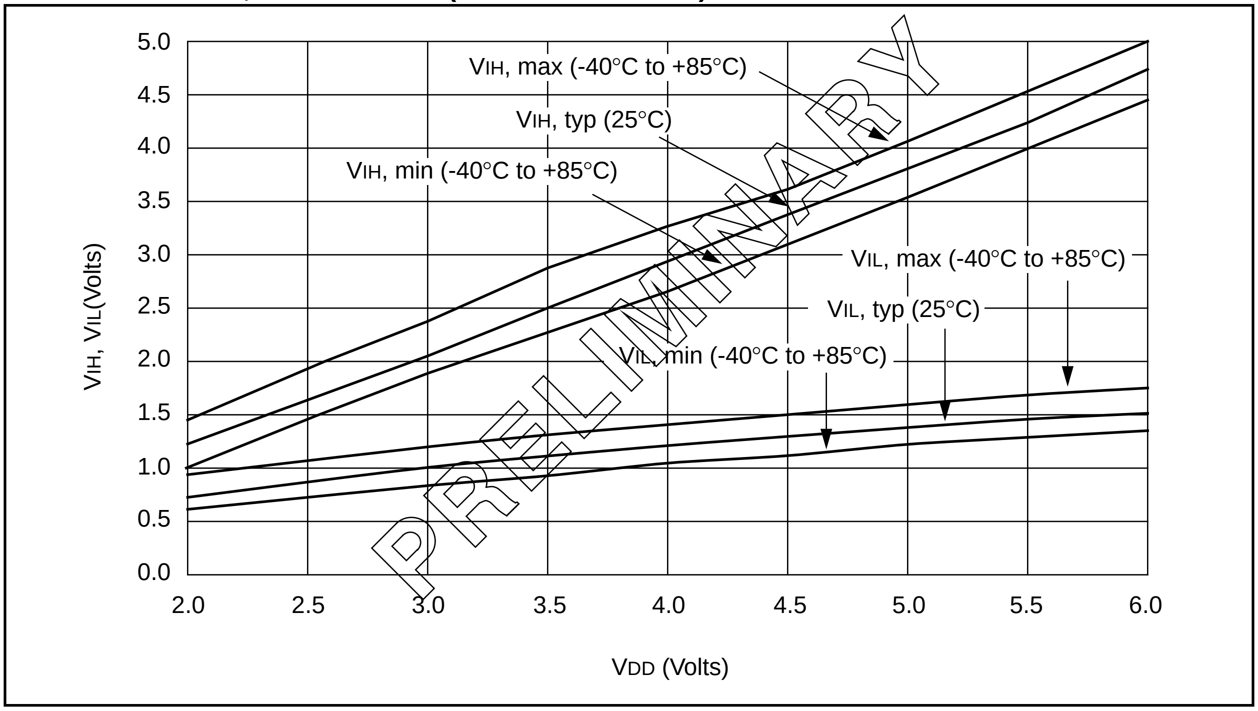
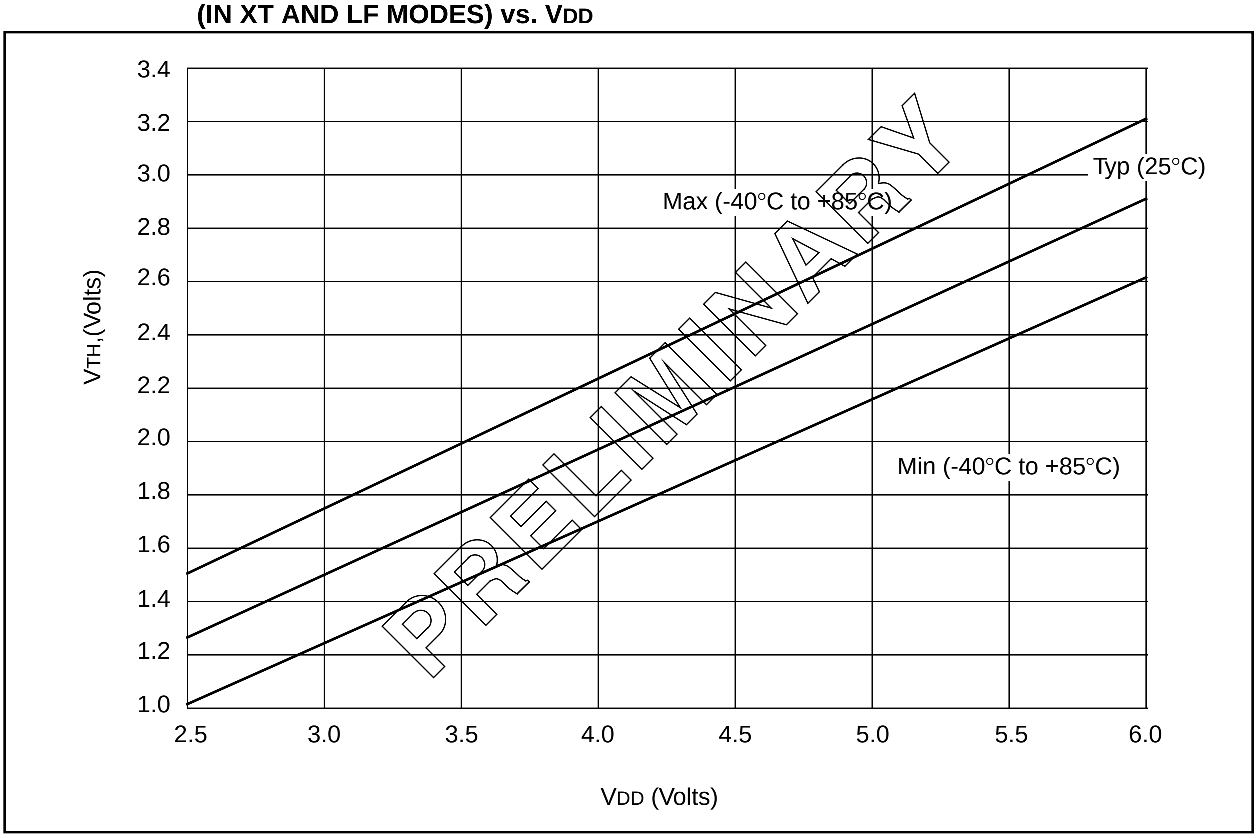


FIGURE 20-20: VTH (INPUT THRESHOLD VOLTAGE) OF OSC1 INPUT (IN XT AND LF MODES) vs. VDD



E.5 PIC16C7X Family of Devices

	Clock	Memory		Peripherals					Features					
	Maximum Frequency of Operation (MHz)	EPROM Program Memory (x14 words)	Data Memory (bytes)	Timer Module(s)	Capture/Compare/PWM Module(s)	Serial Port(s) (SPI/I^{2}C, USART)	Parallel Slave Port	A/D Converter (8-bit) Channels	Interrupt Sources	I/O Pins	Voltage Range (Volts)	In-Circuit Serial Programming	Brown-out Reset	Packages
PIC16C710	20	512	36	TMR0	—	—	—	4	4	13	3.0-6.0	Yes	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C71	20	1K	36	TMR0	—	—	—	4	4	13	3.0-6.0	Yes	—	18-pin DIP, SOIC
PIC16C711	20	1K	68	TMR0	—	—	—	4	4	13	3.0-6.0	Yes	Yes	18-pin DIP, SOIC; 20-pin SSOP
PIC16C72	20	2K	128	TMR0, TMR1, TMR2	1	SPI/I^{2}C	—	5	8	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC, SSOP
PIC16C73	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I^{2}C, USART	—	5	11	22	3.0-6.0	Yes	—	28-pin SDIP, SOIC
PIC16C73A[(1)]	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I^{2}C, USART	—	5	11	22	2.5-6.0	Yes	Yes	28-pin SDIP, SOIC
PIC16C74	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I^{2}C, USART	Yes	8	12	33	3.0-6.0	Yes	—	40-pin DIP; 44-pin PLCC, MQFP
PIC16C74A[(1)]	20	4K	192	TMR0, TMR1, TMR2	2	SPI/I^{2}C, USART	Yes	8	12	33	2.5-6.0	Yes	Yes	40-pin DIP; 44-pin PLCC, MQFP, TQFP

All PIC16/17 Family devices have Power-on Reset, selectable Watchdog Timer, selectable code protect and high I/O current capability.

All PIC16C7X Family devices use serial programming with clock pin RB6 and data pin RB7.

Note 1: Please contact your local sales office for availability of these devices.

INDEX

A

B

C

D

S

T